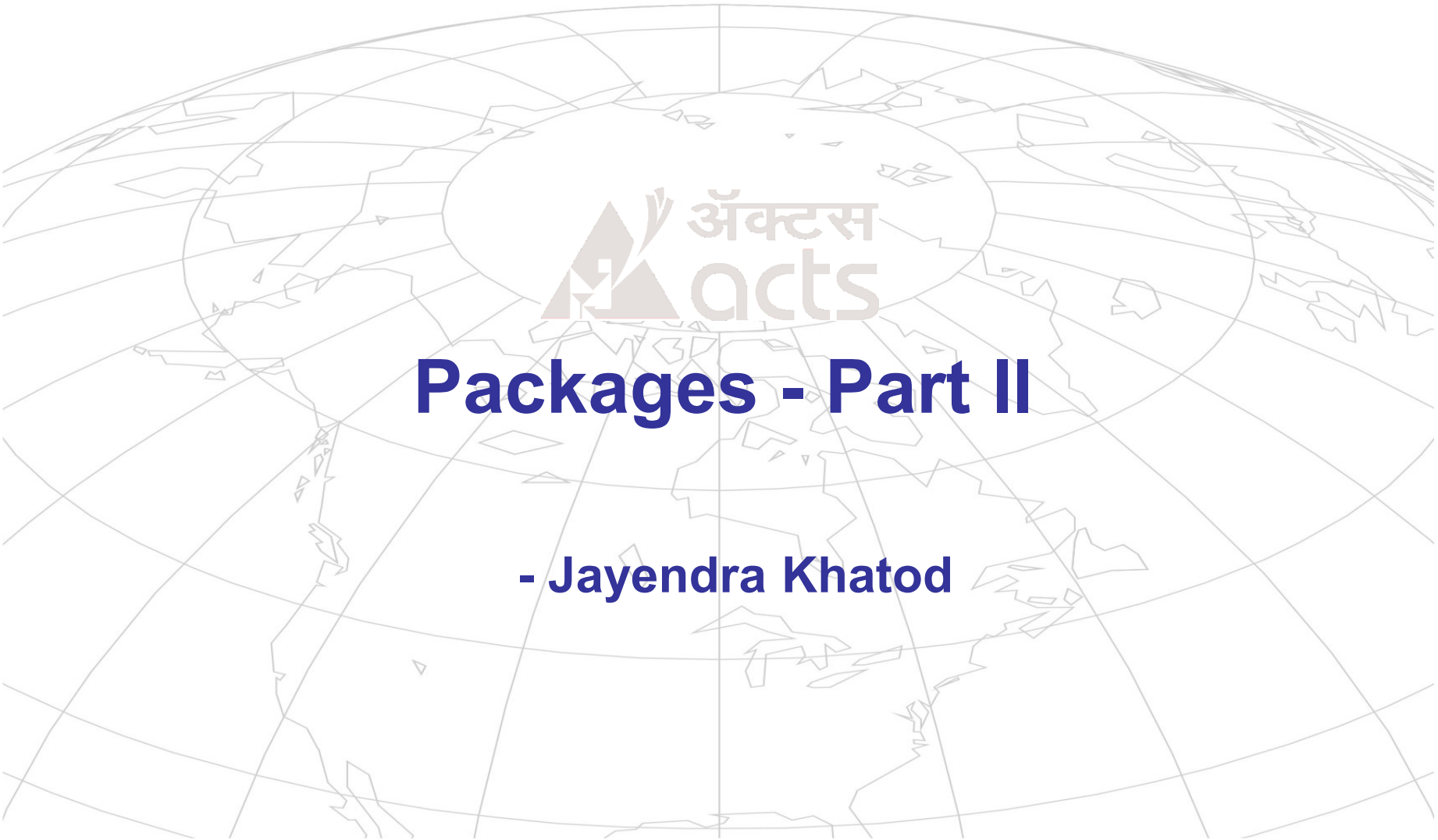


Packages - Part II

- Jayendra Khatod

Objectives

- Practical exercise to create package specification and body
- Write packages that use the overloading feature
- Use Forward Declarations to avoid errors with mutually referential subprograms
- Initialize variables with a one-time-only procedure
- Invoking Package Function from SQL
- Understand persistent states

Creating a Package Specification: Example

```
CREATE OR REPLACE PACKAGE comm_package
 IS
   g_comm NUMBER := 0.10; --initialized to 0.10
  PROCEDURE reset_comm
  (p_comm IN NUMBER);
END comm_package;
/
```

- G_COMM is a global variable and is initialized to 0.10.
- RESET_COMM is a public procedure that is implemented in the package body.

Creating a Package Body: Example

```
CREATE OR REPLACE PACKAGE BODY comm_package
IS
     FUNCTION validate_comm (p_comm IN NUMBER)
     RETURN BOOLEAN
      IS
     v_max_comm NUMBER;
    BEGIN
      SELECT MAX(comm)
      INTO v_max_comm
      FROM emp;
      IF p_comm > v_max_comm THEN RETURN(FALSE);
          ELSE RETURN(TRUE);
      END IF;
    END validate_comm;
...
```

Creating a Package Body: Example

```
   PROCEDURE reset_comm (p_comm IN NUMBER)
   IS
   BEGIN
   IF validate_comm(p_comm)
   THEN g_comm:=p_comm; --reset global variable
   ELSE
   RAISE_APPLICATION_ERROR(-20210,'Invalid
    commission');
   END IF;
   END reset_comm;
END comm_package;
/
```

Declaring a Bodiless Package

```
CREATE OR REPLACE PACKAGE global_vars IS
    mile_2_kilo CONSTANT NUMBER := 1.6093;
    kilo_2_mile CONSTANT NUMBER := 0.6214;
    yard_2_meter CONSTANT NUMBER := 0.9144;
    meter_2_yard CONSTANT NUMBER := 1.0936;
END global_vars;
/
EXECUTE DBMS_OUTPUT.PUT_LINE('20 miles = '||20*
global_vars.mile_2_kilo||' km')
```

Overloading

- Allows you to use the same name for different subprograms inside a package
- Requires the formal parameters of the subprograms to differ in number, order, or datatype family
- Allows you to build more flexibility. A user or application is not restricted by the specific datatype or number of formal parameters
- Restriction: Only local or packaged subprograms can be overloaded

Overloading: Example

```
CREATE OR REPLACE PACKAGE BODY over_pack
IS
  PROCEDURE add_dept
    (v_deptno   IN   dept.deptno%TYPE,
     v_name     IN   dept.dname%TYPE  DEFAULT 'unknown',
     v_loc      IN   dept.loc%TYPE    DEFAULT 'unknown')
  IS
  BEGIN
    INSERT INTO dept
    VALUES   (v_deptno,v_name,v_loc);
  END add_dept;
  PROCEDURE add_dept
    (v_name     IN  dept.dname%TYPE   DEFAULT 'unknown',
     v_loc      IN  dept.loc%TYPE     DEFAULT 'unknown')
  IS
  BEGIN
    INSERT INTO dept
    VALUES   (dept_deptno.NEXTVAL,v_name,v_loc);
  END add_dept;
END over_pack;
```

Forward Declarations

Identifiers must be declared before referencing them.

```
CREATE OR REPLACE PACKAGE BODY forward_pack
IS

   PROCEDURE award_bonus(. . .)
   IS
   BEGIN
    calc_rating(. . .);  --Illegal reference
   END;

   PROCEDURE calc_rating(. . .)
   IS
   BEGIN

   END;
END forward_pack;
```

Forward Declarations

```
CREATE OR REPLACE PACKAGE BODY forward_pack
IS
  PROCEDURE calc_rating(. . .);
      -- forward declaration

 PROCEDURE award_bonus(. . .)
 IS                  -- subprograms defined in
 BEGIN               -- alphabetical order
  calc_rating(. . .);
 . . .
 END;

 PROCEDURE calc_rating(. . .)
 IS
 BEGIN
  . . .
 END;
END forward_pack;
```

Creating a One-Time-Only Procedure

```
CREATE OR REPLACE PACKAGE taxes
IS
   tax   NUMBER;
  ...  -- declare all public procedures/functions
END  taxes;

CREATE OR REPLACE PACKAGE BODY taxes
IS
  ... -- declare all private variables
  ... -- define public/private
procedures/functions
BEGIN
   SELECT    rate_value
   INTO      tax
   FROM      tax_rates
   WHERE     rate_name = 'TAX';
END taxes;
```

Invoke a Package Function from SQL Statements

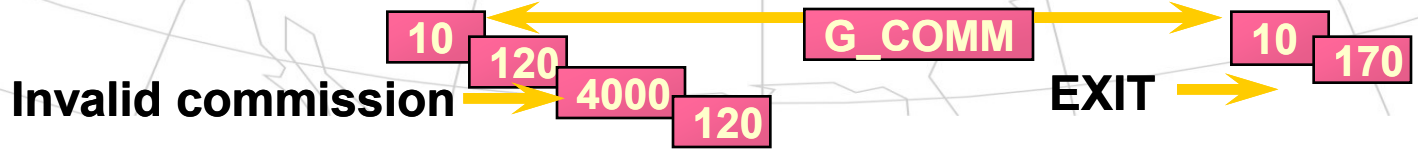
```
SELECT taxes_pack.tax(sal), sal, ename
FROM   emp;


TAXES_PACK.TAX(SAL)          SAL ENAME
-------------------- ---------- ----------
                 400       5000 KING
                 228       2850 BLAKE
                 196       2450 CLARK
                 238       2975 JONES
                 100       1250 MARTIN
                 128       1600 ALLEN
                 120       1500 TURNER

7 rows selected.
```

Persistent State: Package Variables

Time	Scott	Jones
09:00	EXECUTE comm_package.reset_comm(120)	
09:30		INSERT INTO emp (ename,comm) VALUES ('Madonna',2000);
09:35		EXECUTE comm_package.reset_comm(170)
10:00	EXECUTE comm_package.reset_comm(4000)	
11:00		ROLLBACK;
11:01		EXIT;



Persistent State: Package Cursor

Example

```
CREATE OR REPLACE PACKAGE pack_cur
IS
  CURSOR c1 IS SELECT empno
                FROM emp
                ORDER BY empno DESC;
  PROCEDURE proc1_3rows;
  PROCEDURE proc4_6rows;
END pack_cur;
/
```

Persistent State: Package Cursor

```
CREATE OR REPLACE PACKAGE BODY pack_cur
IS
     v_empno NUMBER;

     PROCEDURE proc1_3rows
     IS
     BEGIN
       OPEN c1;
       LOOP
          FETCH c1 INTO v_empno;
          DBMS_OUTPUT.PUT_LINE('Id :'
          ||(v_empno));
          EXIT WHEN c1%ROWCOUNT >= 3;
       END LOOP;
     END proc1_3rows;
     ……(continued..)
```

Persistent State: Package Cursor

```
      ……(continued..)

      PROCEDURE proc4_6rows IS
      BEGIN
        LOOP
           FETCH c1 INTO v_empno;
           DBMS_OUTPUT.PUT_LINE('Id :'
           ||(v_empno));
           EXIT WHEN c1%ROWCOUNT >= 6;
        END LOOP;
        CLOSE c1;
      END proc4_6rows;

END pack_cur;
```

Persistent State: Package Cursor

```
SQL> SET SERVEROUTPUT ON
SQL> EXECUTE pack_cur.proc1_3rows
        Id : 7934
        Id : 7902
        Id : 7900
SQL> EXECUTE pack_cur.proc4_6rows
        Id : 7876
        Id : 7844
        Id : 7839
```

Summary

- Overloading
- Forward referencing
- One-time-only procedures
- Persistent state of packaged variables
- Invoking Package Function from SQL

Thank You !